INPUT Syntax 2 - Using Predefined Map Layout

This form of the `INPUT` statement is used to perform input processing using a map layout that has been created using the Natural map editor.

Map layouts can be used in two ways:

- the program does not provide a parameter list;
- the program does provide a parameter list (*operand1*).

```
INPUT [WINDOW='window-name'] [WITH-TEXT-option]
  [MARK-option]
  [ALARM-option]
  [USING] MAP map-name [NO ERASE]
┌    operand1 ...                  ┐
└    NO PARAMETER                  ┘
```

This chapter covers the following topics:

- INPUT USING MAP without Parameter List
- INPUT Fields Defined in the Program
- INPUT Syntax 2 - Description
- Using the INPUT Statement in Non-Screen Modes
- Processing Data from the Natural Stack

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

INPUT USING MAP without Parameter List

The following requirements must be met when `INPUT USING MAP` is used without parameter list:

- The *map-name* must be specified as an alphanumeric constant (up to 8 characters).
- The map used in this manner must have been created prior to the compilation of the program which references the map.
- The names of the fields to be processed are taken dynamically from the map source definition at compilation time. The field names used in both program and map must be identical.

- All fields to be referenced in the `INPUT` statement must be accessible at that point.
- In structured mode, fields must have been previously defined (database fields must be properly referenced to processing loops or views).
- In reporting mode, user-defined variables may be newly defined in the map.
- When the map layout is changed, the programs using the map need not be recataloged. However, when array structures or names, formats/lengths of fields are changed, or fields are added/deleted in the map, the programs using the map must be recataloged.
- The map source must be available at program compilation; otherwise the `INPUT USING MAP` statement cannot be compiled.

  Note:
  If you wish to compile the program even if the map is not yet available, specify `NO PARAMETER`: the `INPUT USING MAP` can then be compiled even if the map is not yet available.

INPUT Fields Defined in the Program

By specifying the names of the fields to be processed within the program (*operand1*), it is possible to have the names of the fields in the program differ from the names of the fields in the map.

The sequence of fields in the program must match the map sequence. Please note that the map editor sorts the fields as specified in the map in alphabetical order by field name. For more information, see the map editor description in your Natural Editors documentation.

The program editor line command `.I(`*mapname*`)` can be used to obtain a complete `INPUT USING MAP` statement with a parameter list derived from the fields defined in the specified map.

When the layout of the map is changed, the program using the map need not be recataloged. However, when field names, field formats/lengths, or array structures in the map are changed or fields are added or deleted in the map, the program must be recataloged.

A check is made at execution time to ensure that the format and length of the fields as specified in the program match the fields as specified in the map. If both layouts do not agree, an error message is produced.

INPUT Syntax 2 - Description

Operand Definition Table:

Operand	Possible Structure					Possible Formats													Referencing Permitted	Dynamic Definition
map-name	C	S				A	U												yes	no
operand1		S	A			A	U	N	P	I	F	B	D	T	L	C			yes	yes

Syntax Element Description:

Syntax Element	Description
`INPUT WINDOW='window-name'`	**INPUT WINDOW='*window-name*' Option:** This option is described under *Syntax 1* of the `INPUT` statement.
`WITH TEXT/MARK/ALARM-options`	**WITH TEXT/MARK/ALARM Options:** These options are described under *Syntax 1* of the `INPUT` statement; see *WITH TEXT Option*, *MARK Option*, *ALARM Option*.
`USING MAP` *`map-name`*	**USING MAP Clause:** `USING MAP` invokes a map definition which has been previously stored in a Natural system file using the map editor. The *`map-name`* may be a 1- to 8-character alphanumeric constant or user-defined variable. If a variable is used, it must have been previously defined. The map name may contain an ampersand (&); at execution time, this character will be replaced by the one-character code corresponding to the current value of the Natural system variable `*LANGUAGE`. This feature allows the use of multi-lingual maps. The execution of the `INPUT` statement causes the corresponding map to replace the current contents of the screen, unless the `NO ERASE` option is specified, in which case the map will overlay the current contents of the screen.
`NO ERASE`	**NO ERASE Option:** This option is described under *Syntax 1* of the `INPUT` statement; see `NO ERASE`.
`operand1`	**Field Specification:** A list of database fields and/or user-defined variables. The fields must agree in number, sequence, format, length and (for arrays) number of occurrences with the fields in the referenced map; otherwise, an error occurs. When the content of a database field is modified as a result of `INPUT` processing, only the value as contained in the data area is modified. Appropriate database `UPDATE` / `STORE` statements must be used to change the content of the database.

Using the INPUT Statement in Non-Screen Modes

You can change the input mode with the session parameter IM.

In Forms Mode

In forms mode (profile/session parameter IM=F), Natural will display all output text of the map layout on the terminal field by field according to the positioning parameters. This permits the user to enter data on a field by field basis. When all data are entered, the hardcopy output is produced exactly as it would have appeared on the screen.

In forms mode, entering %R permits the operator to retype the entire form in case of an error. The input is processed as in the first execution of the INPUT statement.

In Keyword/Delimiter Mode

In keyword/delimiter mode (profile/session parameter IM=D), data can be entered using keywords or positional input values.

Using keyword input, the terminal operator may enter data for the individual fields using the prompting text that, in forms mode, would have been displayed before the value as a keyword to identify the field. The keyword must be followed by the input assign character (IA parameter), followed immediately by the data. Any spaces following the assign character are taken as data up to the delimiter character (ID parameter). A delimiter character is not required after the last data element. Keyword data for the different fields may be entered in any order separated by the delimiter character. If the operator types in a keyword which is not defined in the INPUT statement, an error message will be returned. Data need not be entered for all input fields. Fields for which no data are entered are set to blank for alphanumeric fields and zero for numeric and hexadecimal fields.

Using positional value input, the terminal operator enters only data for all input fields separated by the currently defined input delimiter character (ID parameter). The sequence of fields for input must correspond to the sequence of the fields in the INPUT statement.

The user may switch from positional to keyword input by entering a number of values in positional input separated by the delimiter character and then switching to keyword mode for selected fields by specifying keywords in front of the values.

After a keyword has been used to position to a field, any non-keyword input following the keyword will be processed as positional input to be assigned to fields following the previously selected field in the INPUT statement.

Note:
A keyword and the corresponding input field must be on the same logical line. If their aggregate length exceeds the line size, adjust the line size (LS parameter) accordingly so that keyword and field fit onto one line.

Data entered in keyword/delimiter mode are validated as for screen mode. An error message will be returned if an attempt is made to enter more characters than defined for a field.

If the INPUT statement is to be processed in keyword/delimiter mode on a buffered (3270-type) terminal or a workstation, all data to be assigned to one INPUT statement must be entered on one screen. ENTER is only to be used when all data to the INPUT statement have been entered.

Processing Data from the Natural Stack

Data elements that have been placed in the Natural stack via a FETCH, RUN or STACK statement will be processed by the next INPUT statement encountered for execution.

The INPUT statement will process the data in keyword/delimiter mode as described above.

If data elements are not available to fill all input fields, fields will be filled with blank/zero depending on the field format. If more data elements are specified than input fields exist, the remaining data are ignored.

When a field is filled with data from the stack, the field attributes do not apply to the data.

The Natural system variable *DATA may be referenced to determine the number of data elements currently available in the Natural stack.